Towards the Establishment of an Interface Protocol between Operating System and Runtime System Software for Exascale Computers

Thomas Sterling
Indiana University
tron@indiana.edu

Andrew Lumsdaine
Indiana University
lums@indiana.edu

Description of Approach

A shift from static to dynamic resource management and task scheduling may be required to ensure sufficient efficiency and scalability of operation for exascale computing. A new system software architecture could combine application runtime system software with innovations in scalable operating systems. Interoperability of these principal software components will require a new system interface and transaction protocol that delivers unprecedented flexibility of mutual support as part of a scalable, resilient, and energy and time efficient execution framework. A Runtime system Interface protocol to the OS (RIOS) is needed to permit a diversity of implementations and policies of both of these major software components while enabling portability across system scales, types, and generations. Such an interface would supersede such prior standards as the POSIX OS API common to many current operating systems.

The proposed approach for this space of exploration is to seek to determine key elements of dialog between the runtime system and operating system. This is a form of co-design although incomplete as it only implicitly incorporates factors associated with user API and underlying hardware architecture while presuming characteristics of both. The proposed approach must iterate, in part, over hypothesized functionality of both software components without determining implementation details. But a resulting solution by its nature sets a division of responsibility between the two while implying a synergy between them. Thus, there are many possible solutions to even a single specified set of requirements.

Summary of Related Work

The majority of runtime work assumes a POSIX OS interface. The use of Catamount and CNK vary although CNK is close to conventional interface.

Assessment of Approach

- Challenges addressed

The new area of opportunity is in the dynamic use of resources that each of the subsystems provide and make available to the other. The interface has to let a scalable operating system work effectively with different runtime systems. It also has to permit a runtime system to serve multiple architecture types through the abstraction of the parallel OS. Broadly challenges include efficiency, scalability, load balancing, data naming and addressing, as well as reliability and energy.

- Maturity

There are many degrees of freedom and this is a period of transition with performance largely determined by efficiency and scalability. Given that performance runtime systems of HPC are the exception rather than the case, it is possible that this area is at its infancy as computing moves from static scheduling to dynamic scheduling and resource management. So conventional practices are mature although always improving. Dynamic runtime scheduling and resource management while there is a long legacy at many levels, within this class of system the focus domain is virgin territory.

- Uniqueness

There is not such an interface protocol between future runtime and OS software modules for Exascale computing.

- Novelty

The experimental ParalleX execution model will be used to drive the initial definition. While ParalleX in its specific definition is distinct, it incorporates a number of mechanisms that are found in prior art and individually employed in other systems (e.g., gas.net).

- Applicability

There are ranges of applications that are regular and static (in schedule and placement) that will receive little benefit in addition to the success of conventional practices. However, for dynamic problems such as adaptive mesh refinement or time varying irregular knowledge management graphs, the information flow between runtime and OS is more important. Also, added challenges of reliability and power management for both classes of problems will be served.

- Effort

An initial interface definition will require a period of functional requirements. A phase of information flow derivation will be evolved to converge on meeting the needs of functionality on each side of the interface. Then an example of a protocol that would meet this structure and an experimental interface specification will be developed. Metrics will be defined to measure the quality of information exchange.